

AMENDMENTS TO THE CLAIMS

1 1. (currently amended) A system for providing a JAVA code release infrastructure with
2 granular code patching, comprising:
3 one or more JAVA code patches, each comprising at least one resource unit, each resource unit
4 comprising metadata and file components;
5 one or more to-be-patched JAVA code libraries, each comprising at least one such resource
6 unit;
7 a patch tool, comprising:
8 a compare module that determines which units in the to-be-patched JAVA code libraries are
9 outdated by comparing the metadata for each such resource unit in the JAVA code
10 patches to the metadata for each such corresponding resource unit in the to-be-patched
11 JAVA code libraries; and
12 a merge module merging each such resource unit in the JAVA code patches into the to-be-
13 patched JAVA code libraries for each such corresponding resource unit that is out-of-
14 date.

1 2. (previously presented) A system according to Claim 1, further comprising:
2 an extract module extracting at least one resource unit from the JAVA_code libraries and
3 modifying one or more JAVA archive files that are out-of-date with the at least one
4 extracted resource unit.

1 3. (previously presented) A system according to Claim 1, further comprising:
2 a sign module signing the JAVA archive files using a digital certificate.

1 4. (previously presented) A system according to Claim 1, wherein the one or more JAVA
2 archive files are modified through at least one of creation, revision or deletion.

1 5. (previously presented) A system according to Claim 1, further comprising:
2 a source repository storing the source file components;
3 a staged patch repository storing the one or more JAVA code patches; and
4 a staged code repository organizing the one or more JAVA code libraries and the JAVA
5 archive files.

1 6. (previously presented) A system according to Claim 1, further comprising:
 2 a resource unit generator processing the file components into at least one such resource unit;
 3 and
 4 a packager packaging at least one such resource unit into one or more of the JAVA code
 5 patches.

1 7. (previously presented) A system according to Claim 6, further comprising:
 2 stored JAVA source code provided as the file components.

1 8. (previously presented) A system according to Claim 7, further comprising:
 2 a compiler compiling at least one JAVA source code file into one or more JAVA classes; and
 3 a resource unit packager module storing the JAVA classes into at least one such resource unit
 4 as the file components.

1 9. (previously presented) A system according to Claim 6, further comprising:
 2 at least one of non-JAVA source and derived code provided as the file components.

1 10. (original) A system according to Claim 6, further comprising:
 2 third party code provided as the file components.

1 11. (original) A system according to Claim 6, further comprising:
 2 a metadata generator generating the metadata for each such resource unit; and
 3 a resource unit packager module storing the generated metadata into the resource unit.

1 12. (original) A system according to Claim 11, wherein the metadata comprises at least one of
 2 a unique identifier and a version attribute.

1 13. (previously presented) A system according to Claim 1, further comprising:
 2 a compare module using a set of rules allowing one of an older resource unit to be replaced by
 3 a newer resource unit and a newer resource unit to be replaced by an older resource unit
 4 to back out a previously-applied JAVA code patch.

1 14. (previously presented) A system according to Claim 1, further comprising:
2 one or more JAVA archive files, each comprising at least one resource unit corresponding to
3 one such resource unit in the JAVA code libraries; and
4 a patch tool referencing JAVA archive file definitions which each correspond to one or more of
5 the JAVA archive files.

1 15. (previously presented) A system according to Claim 14, further comprising:
2 an extract module extracting the resource units from the JAVA code libraries into the JAVA
3 archive files for each such corresponding resource unit that is out-of-date.

1 16. (previously presented) A system according to Claim 15, further comprising:
2 an extract module referencing third party JAVA code libraries not maintained as part of the
3 infrastructure.

1 17. (previously presented) A system according to Claim 1, further comprising:
2 JAVA code libraries implemented as a portable virtual file system which can be used directly
3 by a JAVA Virtual Machine.

1 18. (previously presented) A system according to Claim 1, further comprising:
2 a machine portable infrastructure providing support for JAVA language features by
3 encapsulating JAVA inner classes, nested directory structures, native class names, and
4 native character set.

1 19. (currently amended) A method for providing a JAVA code release infrastructure with
2 granular code patching, comprising:
3 providing one or more JAVA code patches, each comprising at least one resource unit, each
4 resource unit comprising metadata and file components;
5 patching one or more to-be-patched JAVA code libraries, each comprising at least one such
6 resource unit;
7 comparing the metadata for each such resource unit in the JAVA code patches to the metadata
8 for each such corresponding resource unit in the to-be-patched JAVA code libraries;
9 and

10 merging each such resource unit in the JAVA code patches into the to-be-patched JAVA code
11 libraries for each such corresponding resource unit that is out-of-date.

1 20. (previously presented) A method according to Claim 19, further comprising:
2 extracting at least one resource unit from the JAVA code libraries and modifying one or more
3 JAVA archive files that are out-of-date with the at least one extracted resource unit.

1 21. (previously presented) A method according to Claim 19, further comprising:
2 signing the JAVA archive files using a digital certificate.

1 22. (previously presented) A method according to Claim 19, wherein the one or more JAVA
2 archive files are modified through at least one of creating, updating or deleting.

1 23. (previously presented) A method according to Claim 19, further comprising:
2 storing source file components into a source repository;
3 storing one or more JAVA code patches into a staged patch repository; and
4 organizing one or more JAVA code libraries and the JAVA archive files into a staged code
5 repository.

1 24. (previously presented) A method according to Claim 19, further comprising:
2 processing the file components into at least one such resource unit; and
3 packaging at least one such resource unit into one or more of the JAVA code patches.

1 25. (previously presented) A method according to Claim 24, further comprising:
2 providing JAVA source code as the file components.

1 26. (previously presented) A method according to Claim 25, further comprising:
2 compiling at least one JAVA source code file into one or more JAVA classes; and
3 storing the JAVA classes into at least one such resource unit as the file components.

1 27. (previously presented) A method according to Claim 24, further comprising:
2 providing at least one of non-JAVA source and derived code as the file components.

- 1 28. (original) A method according to Claim 24, further comprising:
2 providing third party code as the file components.
- 1 29. (original) A method according to Claim 24, further comprising:
2 generating the metadata for each such resource unit; and
3 storing the generated metadata into the resource unit.
- 1 30. (original) A method according to Claim 29, wherein the metadata comprises at least one of
2 a unique identifier and a version attribute.
- 1 31. (previously presented) A method according to Claim 19, further comprising:
2 using a set of rules to allow one of an older resource unit to be replaced by a newer resource
3 unit and a newer resource unit to be replaced by an older 3 resource unit to back out a
4 previously-applied JAVA code patch.
- 1 32. (previously presented) A method according to Claim 19, further comprising:
2 providing one or more JAVA archive files, each comprising at least one resource unit
3 corresponding to one such resource unit in the JAVA code libraries; and
4 referencing JAVA archive file definitions which each correspond to one or more of the JAVA
5 archive files.
- 1 33. (previously presented) A method according to Claim 32, further comprising:
2 extracting the resource units from the JAVA code libraries for each such corresponding
3 resource unit that is out-of-date.
- 1 34. (previously presented) A method according to Claim 33, further comprising:
2 referencing third party JAVA code libraries not maintained as part of the infrastructure.
- 1 35. (previously presented) A method according to Claim 19, further comprising:
2 implementing JAVA code libraries as a portable virtual file system which can be used directly
3 by a JAVA Virtual Machine.

1 36. (previously presented) A method according to Claim 19, further comprising:
2 providing a machine portable infrastructure supporting JAVA language features by
3 encapsulating JAVA inner classes, nested directory structures, native class names, and
4 native character set.

1 37. (original) A computer-readable storage medium holding code for performing the method of
2 Claim 19.

1 38. (currently amended) A system for patching staged code in a staged JAVA code release
2 infrastructure, comprising:
3 a staged code repository maintaining one or more staged to-be-patched JAVA code libraries,
4 each staged to-be-patched JAVA code library comprising at least one resource unit,
5 each resource unit comprising metadata and file components;
6 a staged patch repository storing one or more JAVA code patches, each JAVA code patch
7 comprising at least one resource unit corresponding to one such resource unit specified
8 in a JAVA code patch definition; and
9 a patch tool accessing one or more JAVA code patches in the staged patch repository,
10 comprising:
11 a compare module comparing the metadata for each resource unit in the JAVA code patches to
12 the metadata in the staged to-be-patched JAVA code libraries for each such
13 corresponding resource unit; and
14 a merge module merging each resource unit in the JAVA code patches into the staged to-be-
15 patched JAVA code libraries for each such corresponding resource unit that is out-of-
16 date.

1 39. (previously presented) A system according to Claim 38, further comprising:
2 an extract module referencing JAVA archive file definitions which each correspond to a staged
3 JAVA archive file, each staged JAVA archive file comprising at least one resource unit
4 corresponding to one such resource unit in the staged JAVA code libraries.

1 40. (previously presented) A system according to Claim 39, further comprising:

an extract module extracting one such resource unit from the staged JAVA code libraries into the staged JAVA archive files for each such corresponding resource unit that is out-of-date.

41. (previously presented) A system according to Claim 40, further comprising:
a sign module creating a digital signature for the staged JAVA archive files using a digital certificate.

42. (currently amended) A method for patching staged code in a JAVA code release infrastructure, comprising:
maintaining one or more staged to-be-patched JAVA code libraries in a staged code repository, each staged to-be-patched JAVA code library comprising at least one resource unit, each resource unit comprising metadata and file components;
accessing one or more JAVA code patches in a staged patch repository, each JAVA code patch comprising at least one resource unit corresponding to one such resource unit specified in a JAVA code patch definition;
comparing the metadata for each resource unit in the JAVA code patches to the metadata in the staged to-be-patched JAVA code libraries for each such corresponding resource unit;
and
merging each resource unit in the JAVA code patches into the staged to-be-patched JAVA code libraries for each such corresponding resource unit that is out-of-date.

43. (previously presented) A method according to Claim 42, further comprising:
referencing JAVA archive file definitions which each correspond to a staged JAVA archive file, each staged JAVA archive file comprising at least one resource unit corresponding to one such resource unit in the staged JAVA code libraries.

44. (previously presented) A method according to Claim 43, further comprising:
extracting one such resource unit from the staged JAVA code libraries into the staged JAVA archive files for each such corresponding resource unit that is out-of-date.

45. (previously presented) A method according to Claim 44, further comprising:

2 creating a digital signature for the staged JAVA archive files using a digital certificate.

1 46. (original) A computer-readable storage medium holding code for performing the method of
2 Claim 42.

47-63. (canceled)

1 64. (new) The system of Claim 1, wherein the compare module resides on a client machine
2 that is separate from a server machine on which resides a patch generator that generated
3 the one or more JAVA code patches.

1 65. (new) The system of Claim 64, wherein the client machine downloads the one or more
2 JAVA code patches from the server machine.

1 66. (new) The method of Claim 19, wherein the step of comparing is performed on a client
2 machine that is separate from a server machine on which resides a patch generator that
3 generated the one or more JAVA code patches.

1 67. (new) The method of Claim 66, further comprising downloading the one or more JAVA
2 code patches from the client machine to the server machine.

1 68. (new) The system of Claim 38, wherein the compare module resides on a client machine
2 that is separate from a server machine on which resides a patch generator that generated
3 the JAVA code patches.

1 69. (new) The method of Claim 42, wherein the step of comparing is performed on a client
2 machine that is separate from a server machine on which resides a patch generator that
3 generated the JAVA code patches.

1 70. (new) A computer-implemented method for patching code, the method comprising:
2 downloading, to a client machine, from a server machine that is separate from the client
3 machine, one or more patches that were generated on the server machine;

4 comparing, at the client machine, (a) contents of the one or more patches to (b) code that is not
5 resident on the server machine;
6 based on the comparing, determining one or more out-of-date portions within the code; and
7 applying at least a part of the one or more patches to the out-of-date portions without applying
8 any part of the one or more patches to parts of the code other than the out-of-date
9 portions.